OpenStreetMap Data Case Study

1. 数据来源

成都市, 四川省, 中华人民共和国
* https://www.openstreetmap.org/relation/2110264
* https://mapzen.com/data/metro-extracts/your-extracts/23b263c2550d
这是我目前所在的城市，我希望通过此次分析对这个城市有更深刻的了解：）

2. 熟悉数据

大致概览数据，发现问题

2.1 提取Sample

为了提高运行效率，先仅从数据提取一个小样本进行测试

```
import xml.etree.cElementTree as ET

OSM_FILE = "Chengdu.osm"
SAMPLE_FILE = "sample.osm"

k = 10 # take every k-th top level element
def get_element(osm_file, tags=("node","way","relation")):
    """
    Yield element if it is the right tag
    :param osm_file:
    :param tags:
    :return:
    """
    context = iter(ET.iterparse(osm_file,events=("start","end")))
    _,root = next(context)
    for event, element in context:
        if event == "end" and element.tag in tags:
            yield element
            root.clear()

with open(SAMPLE_FILE,"wb") as output:
    output.write(b'<?xml version="1.0" encoding="UTF-8"?>\n')
    output.write(b"<osm>\n")

    # Write every kth top level element
    for i, element in enumerate(get_element(OSM_FILE)):
        if i % k == 0:
            output.write(ET.tostring(element,encoding='utf-8'))

    output.write(b"</osm>")
```

2.2 迭代解析

大致看下osm中有多少tag

```
import xml.etree.cElementTree as ET
```

```
OSM_FILE = "Chengdu.osm"
SAMPLE_FILE = "sample.osm"

k = 10 # take every k-th top level element
def get_element(osm_file, tags=("node","way","relation")):
    """
    Yield element if it is the right tag
    :param osm_file:
    :param tags:
    :return:
    """
    context = iter(ET.iterparse(osm_file,events=("start","end")))
    _,root = next(context)
    for event, element in context:
        if event == "end" and element.tag in tags:
            yield element
            root.clear()

with open(SAMPLE_FILE,"wb") as output:
    output.write(b'<?xml version="1.0" encoding="UTF-8"?>\n')
    output.write(b"<osm>\n")

    # Write every kth top level element
    for i, element in enumerate(get_element(OSM_FILE)):
        if i % k == 0:
            output.write(ET.tostring(element,encoding='utf-8'))

    output.write(b"</osm>")
```

output:

```
{'member': 617,
 'nd': 76219,
 'node': 67343,
 'osm': 1,
 'relation': 36,
 'tag': 15143,
 'way': 5685}
```

2.3 标签类型

为了查看数据中的街道有多少问题数据，使用正则表达式对问题数据进行匹配

```
import re
import xml.etree.cElementTree as ET

lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

def key_type(element, keys):
    if element.tag == "tag":
        streetname = element.attrib["k"]
        if re.search(lower, streetname):
            keys["lower"] += 1
        if re.search(lower_colon, streetname):
            keys["lower_colon"] += 1
        if re.search(problemchars, streetname):
            keys["problemchars"] += 1
        else:
            keys["other"] += 1
```

```
    return keys

def process_map(filename):
    keys = {"lower": 0, "lower_colon": 0, "problemchars": 0, "other": 0}
    for event, element in ET.iterparse(filename, events=("start",)):
        keys = key_type(element, keys)
    return keys

print(process_map("sample.osm"))
```

output:

```
{'lower': 14201, 'lower_colon': 887, 'problemchars': 0, 'other': 15143}
```

2.4 探索用户

想看看有多少唯一用户对地图数据有贡献

```
import xml.etree.cElementTree as ET

def process_user(filename):
    users = set()
    for event, element in ET.iterparse(filename, events=("start",)):
        for ele in element.attrib:
            if "uid" in ele:
                if element.attrib["uid"] not in users:
                    users.add(element.attrib["uid"])
    return users

print(len(process_user("Chengdu.osm")))
```

output:

```
658
```

总共有658个唯一用户

2.5 审查街道名

先将样本的街道名打印出来，看看有哪些问题

```
import xml.etree.cElementTree as ET


for event, element in ET.iterparse("sample.osm",events=("start",)):
    if element.tag == "way" or element.tag == "node":
        for ele in element.iter("tag"):
            if ele.attrib["k"] == "addr:street":
                print(ele.attrib["v"])
```

发现如下问题：
1. 部分街道名过于简化，如St, Rd 等
2. 部分街道中文英文名夹杂，如“人民南路四段 - Renminnanlu 4 Duan”
3. 部分街道英文名表述不清，如“Tian Xian Qiao Bei Jie”

3. 数据清理

3.1 街道名

针对上述问题，需要对街道名进行处理，处理方案如下：
1. 将过于简化的字段补充，”St => Street”, “Rd => Street”等
2. 对于中英文夹杂的街道名，仅保留中文名称“人民南路四段 - Renminnanlu 4 Duan => 人民南路四段”
3. 将表述不清的英文街道名修改 “jie => Steet”

```
import xml.etree.cElementTree as ET
from collections import defaultdict
import pprint
import re


OSMFILE = "Chengdu.osm"

street_type_re = re.compile(r'[路段街号道巷]$', re.IGNORECASE)
street_type_en_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square", "Lane",
"Road",
            "Trail", "Parkway", "Commons", "段",
"街","路","号","巷","道","West","East","North","South"]

mapping = {"St":"Street",
           "St.":"Street",
           "st.":"Street",
           " st":" Street",
           "Rd":"Road",
           "Rd.":"Road",
           "Ave":"Avenue",
           "Ave.":"Avenue",
           "jie": "Street",
           "Jie": "Street"
            }

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    n = street_type_en_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)
    else:
        if n:
            street_type = n.group()
            if street_type not in expected:
                street_types[street_type].add(street_name)

def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

def audit(osmfile):
    with open(osmfile,"rb") as f:
        street_types = defaultdict(set)
        for event, elem in ET.iterparse(f, events=("start",)):

            if elem.tag == "way" or elem.tag == "node":
                for tag in elem.iter("tag"):
                    if is_street_name(tag):
                        audit_street_type(street_types, tag.attrib['v'])
        f.close()
```

```
    return street_types

def update_name(name, mapping):
    for key in mapping:
        if "," in name:
            name = name.split(",")[0]
            if name.endswith(key):
                return name.replace(key, mapping[key])
        if "-" in name:
            return name.split(" - ")[0]
        elif name.endswith(key):
            return name.replace(key, mapping[key])


def test():
    st_types = audit(OSMFILE)
    pprint.pprint(dict(st_types))

    for st_type, ways in st_types.items():
        for name in ways:
            better_name = update_name(name, mapping)
            print(name, "=>", better_name)




if __name__ == '__main__':
    test()
```

output:

```
{'Duan': {'人民南路四段 - Renminnanlu 4 Duan'},
 'Jie': {'Jinsi Jie', 'Tian Xian Qiao Bei Jie'},
 'JinAn': {'JinAn'},
 'Lihua': {'Lihua'},
 'Rd': {'Binjiang E Rd'},
 'Shuangliu': {'Yangjiang Rd., Shuangliu'},
 'St': {'Fangchi St', 'Chengshou St'},
 'jie': {'威州镇dong\u2006jie', 'zheng fu jie'},
 'section': {'No. 20 Hongxing road 2 section'},
 'st': {'wenshuyuan st'},
 '人南立交航空路6号丰德国际广场': {'人南立交航空路6号丰德国际广场'},
 '天佑斋': {'天佑斋'},
 '天府大道南延线': {'天府大道南延线'},
 '武侯区郭家桥北街5号附3号(近川大南门).': {'武侯区郭家桥北街5号附3号(近川大南门).'},
 '群光广场': {'锦江区 春熙路 - 群光广场'},
 '高新区南城都汇2A期汇雅园': {'高新区南城都汇2A期汇雅园'}}
锦江区 春熙路 - 群光广场 => 锦江区 春熙路
武侯区郭家桥北街5号附3号(近川大南门). => None
天佑斋 => None
Yangjiang Rd., Shuangliu => Yangjiang Road
Binjiang E Rd => Binjiang E Road
人南立交航空路6号丰德国际广场 => None
Lihua => None
人民南路四段 - Renminnanlu 4 Duan => 人民南路四段
Jinsi Jie => Jinsi Street
Tian Xian Qiao Bei Jie => Tian Xian Qiao Bei Street
wenshuyuan st => wenshuyuan Street
Fangchi St => Fangchi Street
Chengshou St => Chengshou Street
No. 20 Hongxing road 2 section => None
威州镇dong jie => 威州镇dong Street
zheng fu jie => zheng fu Street
天府大道南延线 => None
```

```
JinAn => None
高新区南城都汇2A期汇雅园 => None
```

3.2 将数据转为csv

```
import csv
import codecs
import re
import xml.etree.cElementTree as ET
from collections import defaultdict


OSM_PATH = "Chengdu.osm"

NODES_PATH = "nodes.csv"
NODE_TAGS_PATH = "nodes_tags.csv"
WAYS_PATH = "ways.csv"
WAY_NODES_PATH = "ways_nodes.csv"
WAY_TAGS_PATH = "ways_tags.csv"

LOWER_COLON = re.compile(r'^([a-z]|_)+:([a-z]|_)+')
PROBLEMCHARS = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')


NODE_FIELDS = ['id', 'lat', 'lon', 'user', 'uid', 'version', 'changeset', 'timestamp']
NODE_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_FIELDS = ['id', 'user', 'uid', 'version', 'changeset', 'timestamp']
WAY_TAGS_FIELDS = ['id', 'key', 'value', 'type']
WAY_NODES_FIELDS = ['id', 'node_id', 'position']


street_type_re = re.compile(r'[路段街号道巷]$', re.IGNORECASE)
street_type_en_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)

expected = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Place", "Square",
"Lane","Road","Trail", "Parkway", "Commons",
"段","街","路","号","巷","道","West","East","North","South"]

mapping = {"St":"Street",
           "St.":"Street",
           "st.":"Street",
           " st":" Street",
           "Rd":"Road",
           "Rd.":"Road",
           "Ave":"Avenue",
           "Ave.":"Avenue",
           "jie": "Street",
           "Jie": "Street"
            }

def audit_street_type(street_types, street_name):
    m = street_type_re.search(street_name)
    n = street_type_en_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in expected:
            street_types[street_type].add(street_name)
    else:
        if n:
            street_type = n.group()
            if street_type not in expected:
                street_types[street_type].add(street_name)

def is_street_name(elem):
```

```
    return (elem.attrib['k'] == "addr:street")

def audit(element):
    street_types = defaultdict(set)
    if element.tag == "way" or element.tag == "node":
        for tag in element.iter("tag"):
            if is_street_name(tag):
                 audit_street_type(street_types, tag.attrib['v'])
    return street_types

def update_name(name, mapping):
    for key in mapping:
        if "," in name:
            name = name.split(",")[0]
            if name.endswith(key):
                return name.replace(key, mapping[key])
        if "-" in name:
            return name.split(" - ")[0]
        elif name.endswith(key):
            return name.replace(key, mapping[key])


def shape_tag(element,tag):
    tag = {
        "id": element.attrib['id'],
        "key": tag.attrib['k'],
        "value": tag.attrib['v'],
        "type": 'regular'
    }

    if tag["key"] == "addr:street":
        street_types = audit(element)
        for street_types,ways in street_types.items():
            for name in ways:
                tag["value"] = update_name(name, mapping)
    if LOWER_COLON.match(tag["key"]):
        tag["type"], _, tag["key"] = tag["key"].partition(":")

    return tag

def shape_way_node(element,i,nd):
    return {
        "id": element.attrib['id'],
        "node_id":nd.attrib['ref'],
        "position":i
    }

def shape_element(element, node_attr_fields=NODE_FIELDS, way_attr_fields=WAY_FIELDS,
                  problem_chars=PROBLEMCHARS, default_tag_type="regular"):

    node_attribs = {}
    way_attribs = {}
    way_nodes = []
    tags = []

    tags = [shape_tag(element,t) for t in element.iter("tag")]

    if element.tag == "node":
        node_attribs = {f:element.attrib[f] for f in node_attr_fields}
        return {"node": node_attribs, "node_tags":tags}
    elif element.tag == "way":
        way_attribs = {f:element.attrib[f] for f in way_attr_fields}
        way_nodes = [shape_way_node(element,i,nd) for i,nd in enumerate(element.iter("nd"))]
        return {"way":way_attribs, "way_nodes": way_nodes,"way_tags":tags}

# ================================== #
#         Helper Functions         #
# ================================== #
```

```
def get_element(osm_file, tags=('node', 'way', 'relation')):
    """Yield element if it is the right type of tag"""

    context = ET.iterparse(osm_file, events=('start', 'end'))
    _, root = next(context)
    for event, elem in context:
        if event == 'end' and elem.tag in tags:
            yield elem
            root.clear()



class UnicodeDictWriter(csv.DictWriter, object):
    """Extend csv.DictWriter to handle Unicode input"""

    def writerow(self, row):
        super(UnicodeDictWriter, self).writerow({
            k: (v if isinstance(v,bytes) else v) for k, v in row.items()
        })

    def writerows(self, rows):
        for row in rows:
            self.writerow(row)

# ================================================== #
#               Main Function                        #
# ================================================== #

def process_map(file_in):
    """Iteratively process each XML element and write to csv(s)"""

    with codecs.open(NODES_PATH,'w',encoding="utf-8") as nodes_file, \
            codecs.open(NODE_TAGS_PATH, 'w',encoding="utf-8") as nodes_tags_file, \
            codecs.open(WAYS_PATH, 'w',encoding="utf-8") as ways_file, \
            codecs.open(WAY_NODES_PATH, 'w',encoding="utf-8") as way_nodes_file, \
            codecs.open(WAY_TAGS_PATH, 'w',encoding="utf-8") as way_tags_file:

        nodes_writer = UnicodeDictWriter(nodes_file, NODE_FIELDS)
        node_tags_writer = UnicodeDictWriter(nodes_tags_file, NODE_TAGS_FIELDS)
        ways_writer = UnicodeDictWriter(ways_file, WAY_FIELDS)
        way_nodes_writer = UnicodeDictWriter(way_nodes_file, WAY_NODES_FIELDS)
        way_tags_writer = UnicodeDictWriter(way_tags_file, WAY_TAGS_FIELDS)

        nodes_writer.writeheader()
        node_tags_writer.writeheader()
        ways_writer.writeheader()
        way_nodes_writer.writeheader()
        way_tags_writer.writeheader()


        for element in get_element(file_in, tags=('node', 'way')):
            el = shape_element(element)
            if el:
                if element.tag == 'node':
                    nodes_writer.writerow(el['node'])
                    node_tags_writer.writerows(el['node_tags'])
                elif element.tag == 'way':
                    ways_writer.writerow(el['way'])
                    way_nodes_writer.writerows(el['way_nodes'])
                    way_tags_writer.writerows(el['way_tags'])

if __name__ == '__main__':
    process_map(OSM_PATH)
```

3.3 将数据导入sql

新建数据库Chengdu.db

```
sqlite3 Chengdu.db
```

创建表格

```
CREATE TABLE myTab();
```

导入csv

```
.mode csv
.import my_csv.csv myTab
```

4.探索数据库

4.1 地图中排名前10的邮编

```
import sqlite3
import pandas as pd

db = sqlite3.connect("Chengdu.db")
c = db.cursor()

QUERY = """
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags
UNION ALL
SELECT * FROM ways_tags) tags
WHERE tags.key = "postcode"
GROUP BY tags.value
ORDER BY count DESC
;
"""
c.execute(QUERY)
rows = c.fetchall()
df = pd.DataFrame(rows)
print(df)

db.close()
```

output:

```
         0   1
0   610041  20
1   610000   8
2   610042   4
3   610051   4
4   610081   3
5   610100   3
6   611430   3
7   611731   3
8   610021   2
9   621000   2
10     028   1
11  610031   1
```

```
12  610036   1
13  610061   1
14  610066   1
15  610072   1
16  610084   1
17  610091   1
18  610093   1
19  610105   1
20  610130   1
21  610500   1
22  611130   1
23  611230   1
24  618300   1
```

数据库中出现了共25个邮编，其中028是成都区号不是邮编，621000是绵阳的邮编，需要将这部分去掉

4.2 按城市数量排序

```
QUERY_CITY = """
SELECT tags.value, COUNT(*) as count
FROM (SELECT * FROM nodes_tags UNION ALL
SELECT * FROM ways_tags) tags
WHERE tags.key LIKE '%city'
GROUP BY tags.value
ORDER BY count DESC;
```

output:

```
                  0   1
0                成都  54
1               成都市  38
2           Chengdu  19
3               200  17
4               300   5
5           chengdu   4
6               100   2
7               崇州市   2
8                 2   1
9                 4   1
10               40   1
11               50   1
12           Chengu   1
13           双流县华阳镇   1
14       四川省德阳市广汉市   1
15       四川省绵阳市涪城区   1
16              崇州   1
17     成都市双流区西航港街道   1
18          成都市双流县   1
19  成都市双流县华阳正西街88号   1
20      成都市双流县黄龙大道   1
21         成都市龙泉驿区   1
22            成都郫县   1
23              绵阳   1
24              都江堰   1
```

4.3 数据总览

4.3.1 文件大小

```
Chengdu.osm ......... 129 MB
chengdu.db .......... 69.9 MB
nodes.csv ............. 53.5 MB
nodes_tags.csv ........ 0.97 MB
ways.csv .............. 3.28 MB
ways_tags.csv ......... 3.96 MB
ways_nodes.cv ......... 17.6 MB
```

4.3.2 nodes数量

```
sqlite> SELECT COUNT(*) FROM nodes;
```

output:
673425

4.3.3 ways数量

```
SELECT COUNT(*) FROM ways;
```

output:
56855

4.3.4 unique users 数量

```
SELECT COUNT(DISTINCT(e.uid))
FROM (SELECT uid FROM nodes UNION ALL SELECT uid FROM ways) e;
```

output:
651

4.3.5 前十位贡献者

```
SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
ORDER BY num DESC
```

output:

```
                0       1
0          ff5722  140129
1       katpatuka  106103
2       巴山夜雨   61942
3     AntiEntropy   31173
4         hanchao   28023
5          Nautic   27621
6  geodreieck4711   23458
7         jamesks   19831
8        7thgrade   15423
9      guanchzhou   13383
```

4.3.6 只参与过一次发表的用户数量

```
SELECT COUNT(*)
FROM
(SELECT e.user, COUNT(*) as num
FROM (SELECT user FROM nodes UNION ALL SELECT user FROM ways) e
GROUP BY e.user
HAVING num=1) u;
```

output:

126

结合上面的数据，发现前面两位用户贡献了绝大多数数据，近20%的用户仅参与一次数据发表

4.4 其他的一些探索

4.4.1 出现最多的设施

```
SELECT value, COUNT(*) as num
FROM nodes_tags
WHERE key='amenity'
GROUP BY value
ORDER BY num DESC
LIMIT 10;
```

output:

```
                  0    1
0        restaurant  181
1           toilets   81
2              fuel   73
3           parking   70
4              cafe   64
5              bank   63
6            school   60
7          hospital   46
8   bicycle_parking   30
9  place_of_worship   28
```

餐厅，洗手间，加油站等公共设施出现频率最高

4.4.2 最大的宗教

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE
value='place_of_worship') i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='religion'
GROUP BY nodes_tags.value
ORDER BY num DESC
LIMIT 1;
```

output:

```
          0   1
0  buddhist  13
```

佛教是成都最大的宗教

4.4.3 **最受欢迎的美食**

```
SELECT nodes_tags.value, COUNT(*) as num
FROM nodes_tags
JOIN (SELECT DISTINCT(id) FROM nodes_tags WHERE value='restaurant')
i
ON nodes_tags.id=i.id
WHERE nodes_tags.key='cuisine'
GROUP BY nodes_tags.value
ORDER BY num DESC;
```

output:

```
                              0   1
0                        chinese  12
1                       regional   4
2                          asian   2
3                       barbecue   2
4                       japanese   2
5                            BBQ   1
6                         burger   1
7   burger;italian_pizza;american   1
8                   chinese;local   1
9                 chinese;noodles   1
10                         german   1
11                  international   1
12                          pizza   1
13                        spanish   1
14                    tea;local   1
15                        turkish   1
16                         燃面，姜鸭面   1
```

毫无疑问是中餐

5. 总结

总体来说，这份数据是相对干净的，只有少部分需要进行清理。数据在选取区域时，导致有部分非目标区域数据掺杂其中。还有就是一些街道名称中英文混杂，中英文混用，这部分数据需要进行清理。